In the Specification:

Page 4, paragraph 1, replace as follows:

One method of dealing with bandwidth problems in general has been the use of compression. A lot of work has already been done on lossless data compression (Mark Nelson, The Data Compression Book, M&T Books, 1992). Researchers have developed fast and powerful algorithms for data compression. Their principles are mostly based on Claude Shannon's Information Theory. A consequence of this theory is that a symbol that has a high probability has a low information content and will need fewer bits to encode. In order to compress data well, you need to select models that predict symbols with high probabilities. Huffman coding (Huffman, D.A., "A Method for the Construction of Minimum-redundancy Codes," Proceedings of the IRE, Vol. 40, No. 9, Sept. 1952, pp. 1098-1101) achieves the minimum amount of redundancy possible in a fixed set of variable-length codes. It provides the best approximation for coding symbols when using fixed-width codes. Huffman coding uses a statistical model because it reads and encodes a single symbol at a time using the probability of that character's appearance. A dictionary-based compression scheme uses a different concept. It reads input data and looks for groups of symbols that appear in a dictionary. If a string match is found, a pointer or index into the dictionary can be output instead of the code for the symbol. The longer the match, the better the compression ratio. In LZ77 compression (Ziv et al., "A Universal Algorithm for Sequential Data Compression," IEEE Transaction on Information Theory, Vol. 23, No. 3, May 1997, pp. 337-343), for example, the dictionary consists of all the strings in a window into the previously read input stream. The deflate algorithm (P. Deutsch, "DEFLATE Compressed Data Format Specification version 1.3," RFC 1951, Aladdin Enterprises, May 1996~~, <http://www.ietf.org/rfc/rfc1951.txt>~~) uses a combination of the LZ77 compression and the Huffman coding. It is used in popular compression programs like GZIP (P.

Deutsch, "GZIP File Format Specification Version 4.3," RFC 1952, Aladdin Enterprises, May 1996, <http://www.ietf.org/rfc/rfc1952.txt>) or ZLIB (Deutsch et al., "ZLIB Compressed Data Format Specification Version 3.3," RFC 1950, May 1996, <http://www.ietf.org/rfc/rfc1950.txt>).

Page 6, paragraph 1, replace as follows:

The Wireless Application Protocol Forum (<http://www.wapforum.org>) has proposed an encoding format for XML based on a table (the code space) that matches tokens to XML tags and attribute names ("WAP Binary XML Content Format, <http://www.w3.org/TR/wbxml> "WAP Binary XML Content Format"). It takes advantage both of the offline approach (the code space can be built offline) and of the word-based compression (tags and attribute names are usually the most frequent words in an XML document). Moreover, unlike the previous compression algorithms, it retains the structure of XML documents. But it does not compress at all the character data content nor the attribute values which are not defined in the Document Type Definition (DTD). Moreover, it does not suggest any strategy to build the code space in an efficient way. The preferred encoding format utilized by the present invention addresses both of these drawbacks: it is designed to compress character data and defines a strategy to build code space. The present invention allows for remote procedure calls to be performed utilizing XML-RPC with a reduction in bandwidth utilization.

Page 12, paragraph 1, replace as follows:

The preferred encoding format efficiently represents character data but additional advantages are realized, when it is taken into account that in, for example, typical business to business communications, most of the attribute values are of primitive type like boolean, byte,

integer or float. For example, in a set of typical business to business XML messages provided by the Open Application Group (~~"Open Applications Group," <http://www.openapplications.org>~~ "Open Applications Group"), 70% of the attribute values are of the primitive type. It is inefficient for these attribute values to be transcoded in strings in a binary representation of an XML document. Therefore, the extension codes are used to prefix primitive types like bytes, integers or floats. The following table reminds the meanings given to the global tokens by the WBXML Encoding Specification and also precises the meanings of the extension tokens which have been redefined for the needs of the preferred encoding method (these tokens appear in bold in table 1).

Page 23, paragraph 2, replace as follows:

The preferred encoding format is designed to represent XML documents in a compact way using tokens to represent tags and attributes instead of strings. The previous discussions have been made with regards to a parser which could work directly upon the binary XML format which utilizes the tokens. However, it is possible to decompress the compressed streams and work on them utilizing a conventional parser. While this is the case, however, particular advantages in processing speed are realized if parsers are implemented which work directly upon the compressed stream itself. Preferably, parsers for documents encoded using this format are built implementing the two standard application programming interfaces (API): DOM (~~"Document Object Model (DOM) Level 1 Specification Version 1.0, W3C Recommendation 1," <http://www.w3.org/TR/REC-DOM-Level-1>~~("Document Object Model (DOM) Level 1 Specification Version 1.0, W3C Recommendation 1") and SAX (~~"SAX 1.0: The Simple API for XML," <http://www.megginson.com/SAX/>~~"SAX 1.0: The Simple API for XML").